Initial Design Specification

4SC020 Embedded Motion Control
Tutor : Wouter Kuijpers
Group 3

Jelte Borsboom
Aparnasri Sekar
Nick Peters
Ayisha Wafa

1 Requirements

1. PICO may not touch the walls or any other obstacle in the maze at any time
2. PICO must operate fully autonomous (i.e. without input from the team)
3. PICO may not be idle for more than 30 seconds from the start of the program unless it has passed the exit
4. PICO must be able to detect available space using the Laser Range Finder (LRF) and move to the detected space according to the motion planning algorithm
5. PICO must be able to find the exit in finite time ($\leq$ 7 min for maze and $\leq$ 5 min for corridor.)
6. PICO must be able to detect the objects (dead ends) that have a high probability of being a door
7. PICO must be able to open the door in the maze and pass through it

The overall behavior of the PICO is depicted in figure 1 as a task-skill-motion framework.

2 Specifications

The above mentioned requirements can be implemented by software modules given below:

1. Detect the walls.
2. Move away from the wall
3. Time calculator at standstill.
4. Motions if stand still time exceeds 30 sec.
5. Detect the free spaces.
6. Motions if detected the free space
7. Angle calculator at turns
8. Loop detector
9. Detect the door and ring bell.
10. Motions if door didn't open.
11. Motions at T junction.

3 Functions

3.1 Path-finding Supervisor

Pledge Algorithm: Takes the extracted wall information as input and sets a movement goal for the PICO and move towards it. The algorithm functions as a continuous loop. (Maze challenge)

Cornering: Take the corner if possible from the input of the wall information. (Corridor challenge)

3.2 Door Supervisor

Ring Bell: Ring the bell to open the door

Wait: Wait for 5 sec to for the door opening sequence.

3.3 Wall / Path Detection

Read LRF data: Gets raw data from the LRF sensor

Filter LRF data: Reduces noise from raw LRF data and splits it into 3 directions (left, right and front).

Transform data: Calculate one distance approximation for the 3 directions.

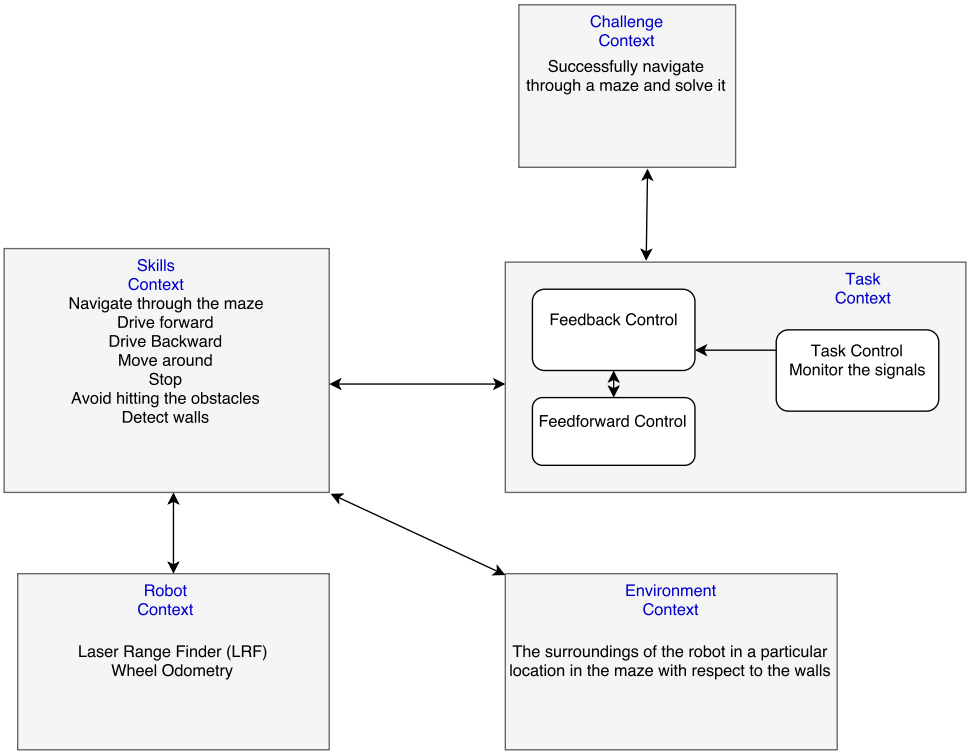

Figure 1: Task-Skill-Motion Framework

Possiblity checker: Calculate for all directions if movement in that direction is possible, taking a 'safe' zone around PICO into account.

3.4 Motion Supervisor

Position stabilizing: Feedback loop that keep PICO approximately in the center of the corridor by implementing a feedback controller for the distance to the walls on the left and right of the robot.

Turn Corner : Make a 90 degree turn

3.5 Actuator Supervisor

Steady PICO: Keep the front of the PICO aligned to the direction of movement.

Omniwheels handeling: Set speed and angle of Omniwheels

4 Interfaces

The interfaces are defined as the movement of data form the functions defined in section 3, it is visialized in figure 2. The different interfaces are discussed and the intented way of inplementation in code is added in italic.

1. Sending information about current state, which is distance to walls at -90 degrees, 0 degrees and 90 degrees from the centerpoint and whether it has potential from moving forward and turning left or right. *Struct with float values and booleans*

2. Forward information about potential movement. *Struct of booleans*

3. Forward information about wall distances. *Struct of floats*

4. Turning command, which is either 0 (turn around), 1 (turning left) or 2(turning right). *Integer*

5. Send information about the possibility of a door. *Boolean*

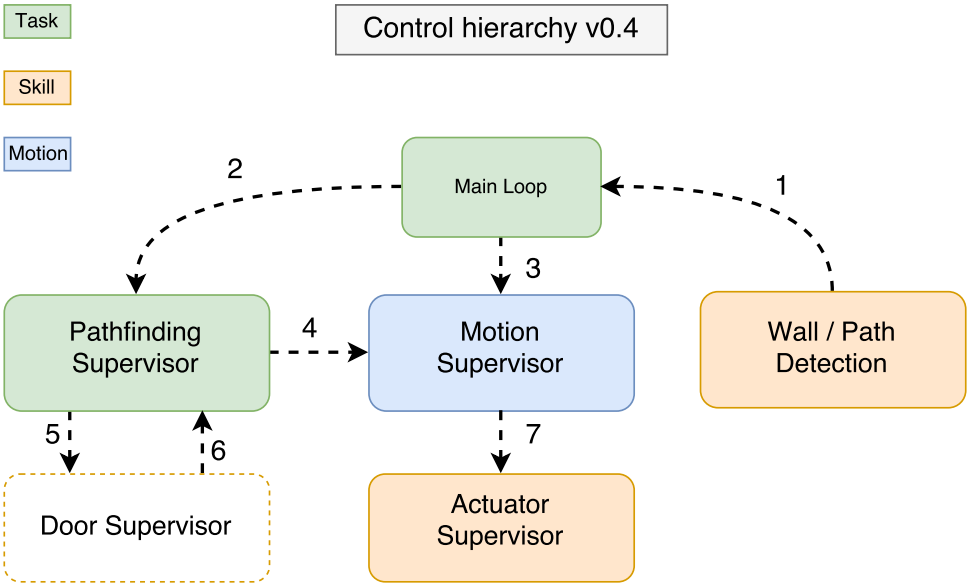

Figure 2: function-Interface diagram

6. Confirmation that door handling is finished. *Boolean*
7. Setting the values for the actuator. *Struct of floats*

5 Components

1. Sensors
   - Laser Range Finder (LRF)
   - Odometry (Wheel Encoder)
2. Holonomic base (omni-wheels): Max. translational speed of 0.5 m/s , Max angular speed of 1.2 rad/s.
3. Embedded platform running Ubuntu 14.04 on an intel i7 (other specifications are unknown)
4. Bell to open door